Applying Advances in Parallel Computing to Pyrolysis Modeling Algorithms

Thomas Dijkmans, Kevin M. Van Geem, Guy B. Marin

Laboratory for Chemical Technology, Ghent University, Ghent, Belgium

Introduction

Detailed modeling of complex kinetic networks still remains a daunting task. The computational cost of a single simulation rises quickly when the kinetic network becomes larger. The latter is in particularly true for combustion, oxidation and pyrolysis mechanisms where considering several thousands of species is no exception. Moreover the computational power of a single core has flattened out during the last years (Figure 1**)**, and hence, the simulation time rises drastically with increasing network sizes. The computational cost of these simulations is specific to the simulation type and determined by a variety of factors such as properties of the flow to be simulated, types of solvers involved, the numerical differentiation scheme, spatial and temporal resolutions and inter-process communication for high-performance simulations[1]. The overall simulation time is often dominated by the evaluation and factorization of the Jacobian matrix and the evaluation of the reaction rates so the optimization and speedup of chemical codes mainly focusses on these parts[2]. In these schemes the Jacobian is often calculated by finite differences. This requires n^2 times the evaluation of the reaction rates per integration step which explains the large computational cost of the evaluation of the reaction rates as well.

The above is mainly true for implicit solvers which are often available in different kinds of commercial code (e.g. CHEMKIN). Explicit solvers often do not require the calculation of the Jacobian matrix which reduces the computational cost of these kind of solvers drastically and the reaction rates only need to be evaluated a limited number of times per integration step. Explicit solvers require however small integration steps to overcome the inherent chemical stiffness of the reaction mechanism. This chemical stiffness is typically induced by QSS (Quasi Steady State) species and can be reduced by using the QSSA (Quasi Steady State Approximation) method. The reduced chemical stiffness allows for larger integration steps to be taken[3-6].

The time distribution in these explicit solvers is for these reasons completely different than those of implicit solvers and only limited efforts have been made to optimize these solvers for newer computer architecture[7]. Moreover, when these explicit solvers are used it is clear that other components, that used only limited amount of time in the implicit scheme, will become very time consuming in an explicit scheme.

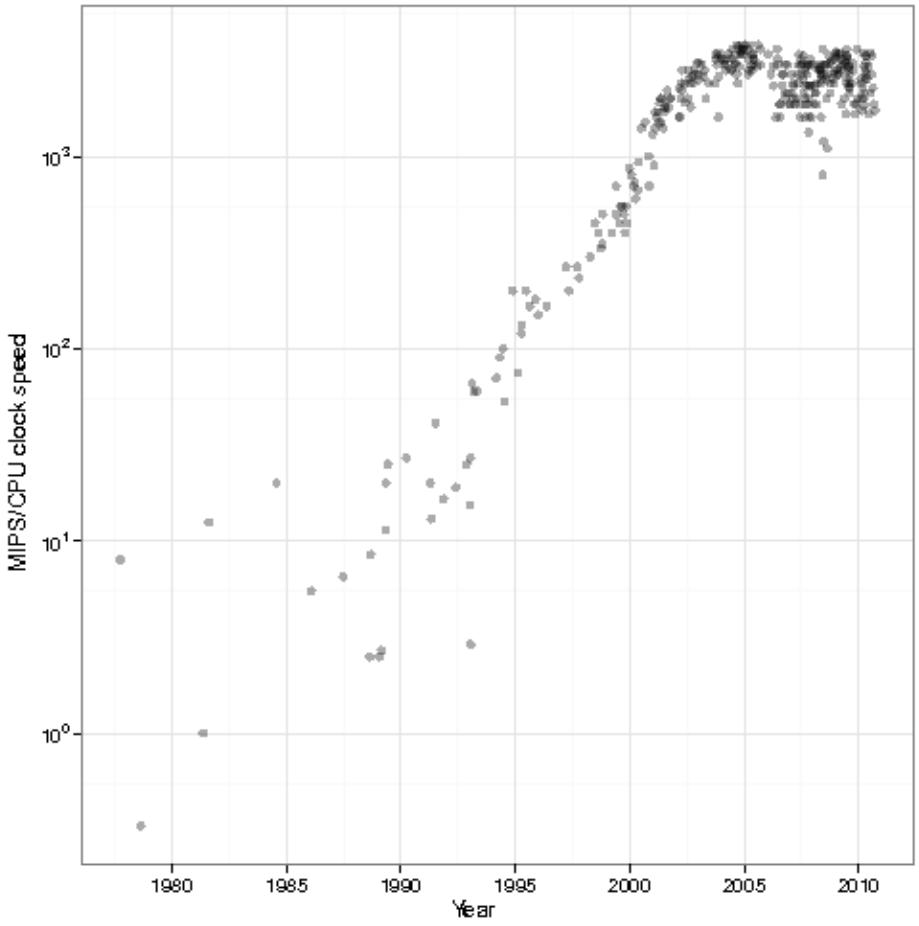

Figure 1: Evolution of single core clock speed over time (MIPS = Mega instructions per second)[8]

To illustrate the possible speed-up for pyrolysis applications using explicit solvers we have applied these advances in parallel computing on free-radical based reaction mechanisms of different sizes and characteristics implemented in COILSIM1D. COILSIM1D combines a recently developed single event microkinetic model (CRACKSIM[9]) for simulating the steam cracking process with a 1D reactor model (Plug flow reactor). COILSIM1D is able to simulate the cracking of a very broad range of industrially relevant feedstocks, from ethane to LPG, over naphtha to gas oil and vacuum gas oils. COILSIM1D solves the differential equations by applying the QSSA to the total sum of radicals present in the mixture. This means that in each integration step the rate of formation of all the radicals is assumed to be equal to the rate of disappearance of all the radicals. In this way the concentration of the radicals can be obtained by solving a set of algebraic equations. This reduces the chemical stiffness of the set of differential equations and makes it possible to solve the differential equations using a Runge-Kutta method. The duration of a single simulation however drastically increases when the feedstocks become more complex due to the increasing size of the reaction mechanism that is considered (see Figure 1). The highly parallel structure of certain parts of the code make COILSIM1D the ideal candidate for hybrid CPU/GPU calculations. Since Coilsim1D is originally written in Fortran the author's preferred using CUDA Fortran from PGI over CUDA C from NVidia. To the author's knowledge this is the first time that CUDA Fortran is applied for combustion or pyrolysis applications.

Programming approach

Identifying time consuming subroutines

The first step in speeding up the code is of course to identify the calculations which are the most time consuming, so-called profiling. Profiling of Coilsim1D is done using three distinct simulations. Case 1 is a simulation where both the temperature and pressure profiles are specified and only the continuity equation needs to be solved. The energy and momentum balances do not need to be solved. Case 2 is a simulation where the heat flux to the gas phase is specified and case 3 is a simulation were the outlet conditions and the shape of the heat flux profile are specified. In both case 2 and case 3 the energy and momentum equations need to be solved as well as the continuity equations.

Figure 2 shows the results of the profiling study. Profiling of Coilsim1D has shown that both the solver and the evaluation of the reaction rates are still among the most costly subroutines even though the model has been simplified to reduce simulation times (see Figure 2). When both temperature and pressure profile are given and only the material balances need to be solved this part of the calculations takes about 85% of the total time (Case 1). However when both energy and momentum balances need to be solved additionally it only uses between 10 and 30% of the total time (Case 2 and 3). In these cases the evaluation of the viscosity of the gas phase becomes the most time consuming calculation taking between 75 and 85% of the total time. The increasing complexity when going from case 1 to case 3 is also clearly visible in the total simulation time which goes from only 0.48s to 23.03s.

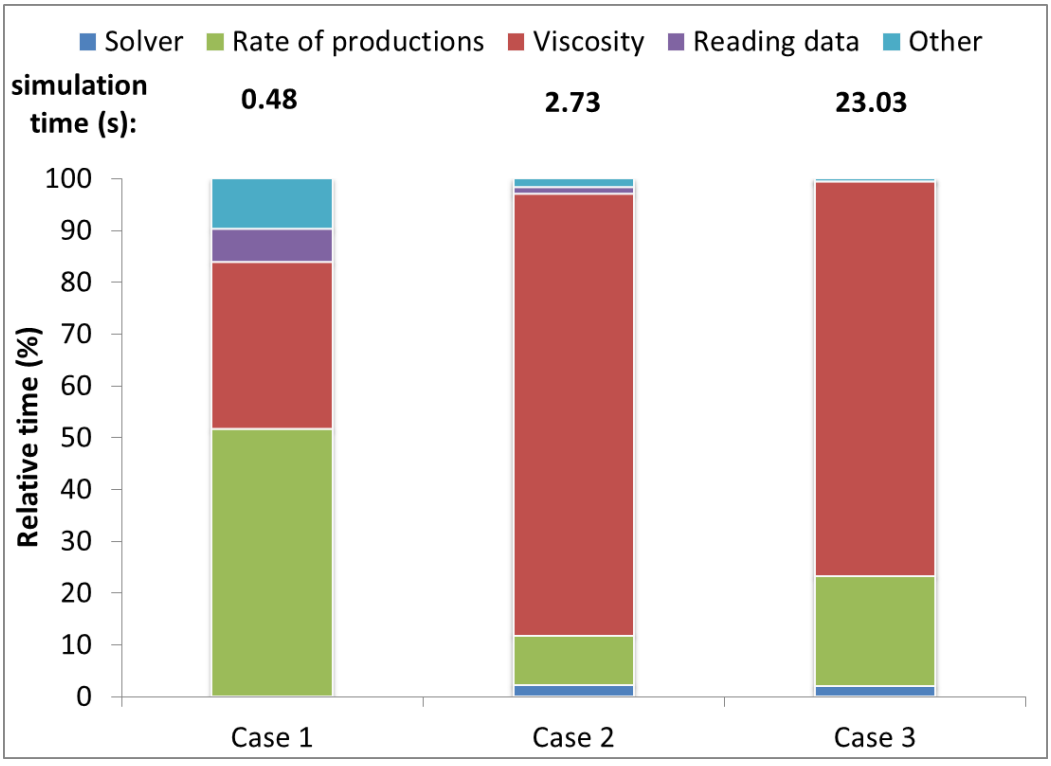

Figure 2: Relative time consumption of different subroutines for CPU calculations using 170 species

Analyzing the viscosity calculations

In case of Coilsim1D the method of Wilke [10] is used to calculate the viscosity of the gas mixture. In the method of Wilke following coefficients needs to be calculated:

$$\phi_{ij} = \frac{[1+(\eta_i/\eta_j)^{1/2}(M_i/M_j)^{1/4}]^2}{[8\,(1+M_i/M_j)]^{1/2}} \qquad \text{[Eq. 1]}$$

These coefficients are dependent on the temperature and the concentrations of the species in the gas phase and need to be re-evaluated for changes in either one of these variables. This coefficient matrix consists of $n_{species} \times n_{species}$ elements. Each element requires a single evaluation of equation 1. It is clear that the calculation of this coefficient matrix is of second-order. Figure 3 shows the total time spent doing viscosity calculations for different network sizes. The curve shows a clear quadratic trend which corresponds to the fact that the algorithm is of second-order.

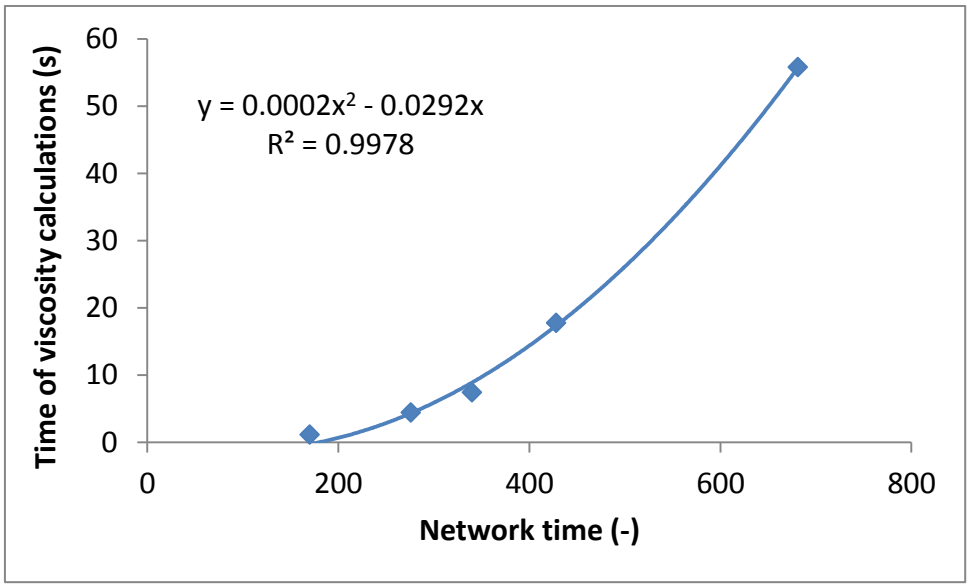

Figure 3: Cumulative duration of viscosity calculations in Coilsim1D for different network sizes

For the evaluation of the coefficients both the molar flow rates and the viscosity of the pure components are needed. The calculation of the viscosity of a pure gas is calculated using the method of Stiel and Thodos [11] while the molar flow rate can easily be derived from the composition of the gas phase. Both calculations are of first-order and will have a limited influence when the size of the reaction network increases.

Based on the relative time consumption of the viscosity calculations a theoretical maximum speedup can be calculated using the following equation

$$Max\ Speedup = \frac{1}{(1 - t_{visc,rel})}$$

With $t_{visc,rel}$ the relative time consumption of the viscosity calculations. This maximum speedup goes from only 0.9 for smaller networks (170 species) to 3.5 for the larger networks (681 species) in COILSIM1D.

Exploiting GPU for viscosity calculations

Equation 1 shows that the evaluation of a single Wilke coefficient is not dependent on previously evaluated Wilke coefficients. This independence make them ideal to be calculated on a GPU. In Figure 5 a comparison is made between the original CPU algorithm of the viscosity subroutine and the modified CPU/GPU version. The main idea behind the calculations remains essentially the same. Several do loops in the original code have been replaced with GPU kernels and since the GPU doesn't have access to the global memory of the machine transfer operations have been added to the new subroutine.

Results

Figure 4 shows the speedup at different network sizes when Coilsim1D is calculated on a hybrid CPU/GPU system. The system is equipped with an Intel Xeon E5620 processor with 6 Gb of memory. It is extended with a Nvidia Tesla C2075 card for the CPU/GPU hybrid calculations. The highest speedup is obtained at the larger network sizes. For example the simulation time for a network with 681 species is reduced from 72.4s to 20.8s when the hybrid CPU/GPU version is used. For smaller network sizes the speedup is negligible or even lower than one. This is caused by an increased relative importance of the copy operation from the CPU memory to the GPU memory and vice versa. Figure 4 also shows the theoretical maximum speedup of the program. At large network sizes our algorithm closely approximates this maximum so additional speedup of the program by further optimizing the viscosity calculations will be difficult if not impossible to obtain.

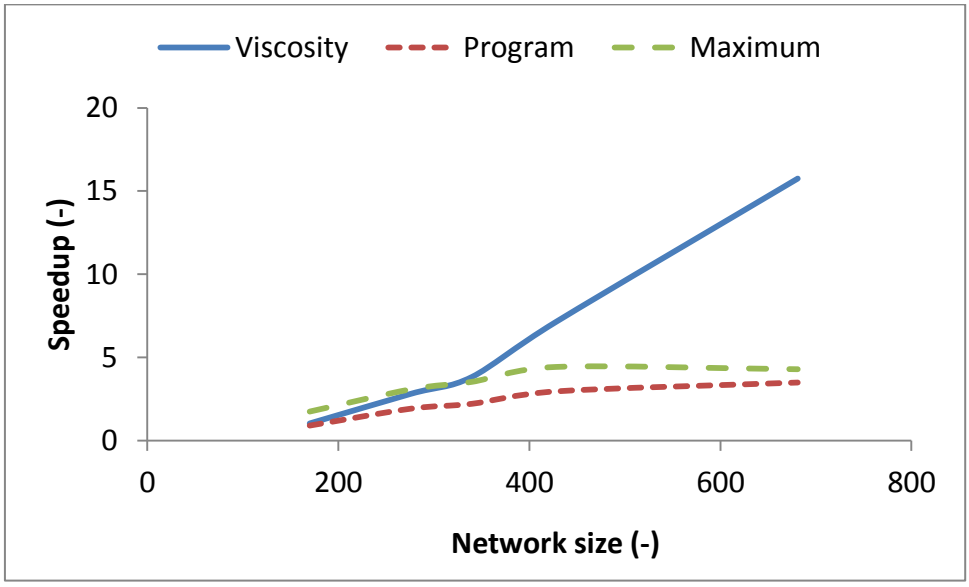

Figure 4: Speedup of Coilsim1D when comparing a CPU simulation with a CPU/GPU hybrid simulation

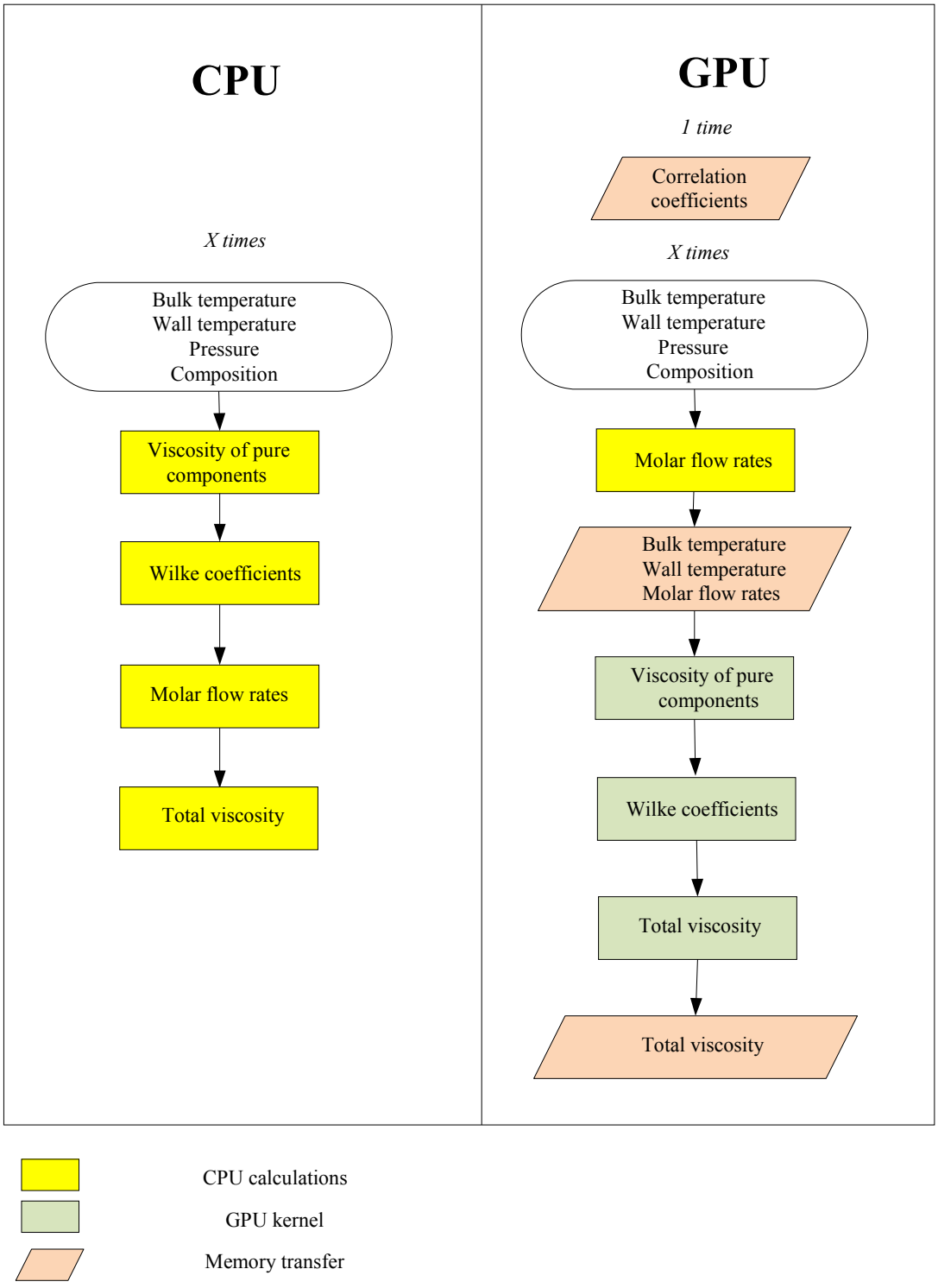

Figure 5: Flow sheet of the calculations in the physical property subroutine. Left: CPU version. Right: CPU/GPU version

Conclusions

Depending on the type of solver used to solve the differential and algebraic equations, different routines inside the simulation code can play a major role in the simulation cost. In case of implicit solver a lot of time is consumed to evaluate the Jacobian matrix by finite differences. Each evaluation of the Jacobian requires that the reaction rates are calculated n² times. Explicit solvers often do not require the evaluation of the Jacobian and the simulation time of these solvers is drastically lower than that of implicit solvers. This article showed that even for this explicit solvers significant speedup can be obtained by using a GPU to do part of the calculations. This is because other subroutines become relatively more time consuming than the evaluation of the Jacobian and the rates. A speedup of 3.5 was obtained for the larger networks which contain 681 species by adapting the viscosity calculations so that the viscosity can be calculated on a GPU.

Aknowledgments

This work was supported by the company SABIC.

References

1. Lu T, Law CK. Toward accommodating realistic fuel chemistry in large-scale computations. *Progress in Energy and Combustion Science.* 2009;35(2):192-215.

2. Shi Y, Green Jr WH, Wong H-W, Oluwole OO. Redesigning combustion modeling algorithms for the Graphics Processing Unit (GPU): Chemical kinetic rate evaluation and ordinary differential equation integration. *Combust. Flame.* 2011;158(5):836-847.

3. Odman MT, Kumar N, Russell AG. A comparison of fast chemical kinetic solvers for air quality modeling. *Atmospheric Environment. Part A. General Topics.* 1992;26(9):1783-1789.

4. Djouad R, Sportisse B. Solving reduced chemical models in air pollution modelling. *Applied Numerical Mathematics.* 2003;44(1–2):49-61.

5. Kovács T, Zsély IG, Kramarics Á, Turányi T. Kinetic analysis of mechanisms of complex pyrolytic reactions. *Journal of Analytical and Applied Pyrolysis.* 2007;79(1–2):252-258.

6. Clymans PJ, Froment GF. Computer-generation of reaction paths and rate-equations in the thermal-cracking of normal and branched paraffins. *Comput. Chem. Eng.* 1984;8(2):137-142.

7. Shi Y, Green WH, Wong HW, Oluwole OO. Accelerating multi-dimensional combustion simulations using GPU and hybrid explicit/implicit ODE integration. *Combust. Flame.* Jul 2012;159(7):2388-2397.

8. Gillespie C. CPU and GPU trends over time. 2011. http://www.r-bloggers.com/cpu-and-gpu-trends-over-time/. Accessed 6 september 2012.

9. Van Geem KM, Heynderickx GJ, Marin GB. Effect of radial temperature profiles on yields in steam cracking. *Aiche J.* Jan 2004;50(1):173-183.

10. Reid R.C. PJM, Poling B.R. *Properties of gases and liquids*: McGraw-Hill; 1979.

11. Jossi JA, Stiel LI, Thodos G. The viscosity of pure substances in the dense gaseous and liquid phases. *Aiche J.* 1962;8(1):59-63.